# Determining the Quality of Mathematical Software Using Reference Data Sets

Keith J. Lines, Ian M. Smith

Data Science Group,
National Physical Laboratory,
Teddington, TW11 0LW, UK
keith.lines@npl.co.uk, ian.smith@npl.co.uk

**Abstract**

This paper describes a methodology for evaluating the numerical accuracy of software that performs mathematical calculations. The authors explain how this methodology extends the concept of *metrological traceability*, which is fundamental to measurement, to include software quality.

Overviews of two European Union-funded projects are also presented. The first project developed an infrastructure to allow software to be verified by testing, via the internet, using reference data sets. The primary focus of the project was software used within systems that make physical measurements. The second project, currently underway, explores using this infrastructure to verify mathematical software used within general scientific and engineering disciplines.

Publications on using reference data sets for the verification of mathematical software are usually intended for a readership specialising in measurement science or mathematics. This paper is aimed at a more general readership, in particular software quality specialists and computer scientists. Further engagement with experts in these disciplines will be helpful to the continued development of this application of software quality.

**Keywords:** Software, Standards, Traceability, TraCIM, ValTraC, Verification

## 1.0 Introduction

National Measurement Institutes (NMIs), such as the National Physical Laboratory (NPL) [1] in the UK and the Physikalisch-Technische Bundesanstalt (PTB) [2] in Germany, have developed methodologies to evaluate quality criteria, such as

numerical accuracy, of software implementations of mathematical algorithms (hereafter referred to as mathematical software).

Mathematical software is increasingly essential to modern-day measurement systems. Quantifying the effect software quality has on the accuracy of the measurement results provided by such systems is an important activity for NMIs and industry. The demand for ever more accurate measurements, supported by ever more complex software, can only increase [3].

In the following paper the authors present an overview of a methodology to evaluate the numerical accuracy of mathematical software. This methodology uses reference data sets, sometimes known as *numerical artefacts*. These artefacts are analogous to the physical artefacts with which NMIs establish the *metrological traceability* of measurements of physical quantities such as mass.

Sections 2 and 3 provide background information, including an introduction to metrological traceability. Section 4 provides an overview of the methodology. As well as being used for verification of mathematical software within measurement systems, the methodology can be applied to mathematical software more generally [4]. Section 5 explains how the concept of metrological traceability can be extended to the verification of mathematical software.

Coordinate measuring machines (CMMs) [5] are described in section 6. The verification of mathematical software within these devices is a major area of application for the methodology.

Section 7 contains an overview of the European Union-funded project TraCIM [6]. Amongst other deliverables the project developed services that allow reference data sets to be used, via the internet, directly in devices running the software to be verified [7]. Section 8 describes another European Union-funded project, ValTraC [8], that is exploring the use of the TraCIM system within general scientific and engineering disciplines. The authors end with some thoughts on future directions for this work.

# 2.0 Verification and Validation

This paper uses the definitions of verification and validation provided in ISO/IEC/IEEE 24765:2010 [9]:

Table 1. Terminology

| | |
|---|---|
| Verification | Formal proof of program correctness. |
| Validation | Confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled. |

Therefore by verification the authors mean "Has the mathematics been implemented correctly?" and by validation, "Has the implementation of the mathematics met a user requirement?

# 3.0 Metrological Traceability

Metrological traceability is a fundamental concept within measurement science (or metrology). The International Bureau of Weights and Measures (BIPM) [10] is the body with responsibility for ensuring and promoting the global comparability of measurements via the International System of Units (SI). In its International Vocabulary of Metrology (VIM) [11], BIPM defines metrological traceability as:

> "Property of a measurement result whereby the result can be related to a reference through a documented unbroken chain of calibrations, each contributing to the measurement uncertainty"

Metrological traceability ensures that a kilogram of potatoes purchased from a food retailer in the UK must (subject to measurement uncertainty) have the same mass as the UK's national standard kilogram held at NPL. The retailer, trading standards officers, etc. must use weighing devices calibrated in an unbroken *traceability chain* ending at NPL, and NPL must in turn be traceable to the BIPM.

International intercomparisons between NMIs demonstrate that the participants' measurement results are equivalent and therefore verify the way they maintain and disseminate their national mass standards. Such intercomparisons are also subject to measurement uncertainties. A discussion of measurement uncertainty is beyond the scope of this paper. Reference [12] provides an introduction to the topic.

Figure 1 provides a simple example of a traceability chain.

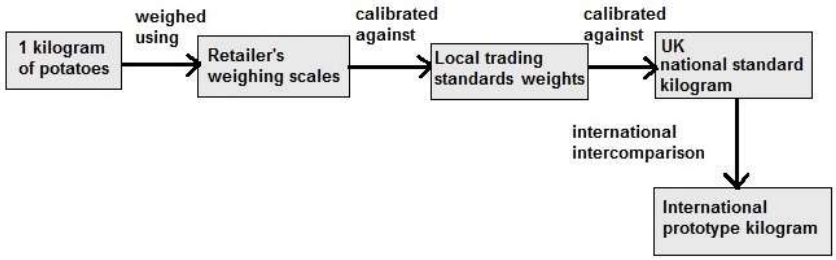


Figure 1:Example of a traceability chain

As will be described in the section 5, the concept of metrological traceability can be extended to verification of mathematical software. But first, an explanation of some further terminology is required.

# 4.0 Verification of Mathematical Software

The following section provides an introduction to the methodology. It begins with a summary that introduces the concepts of *computational aims* and *reference data sets*.

## 4.1 Summary

The methodology described in this paper can be summarised as follows:

1. Provide a clear, complete and unambiguous statement of the mathematics to be implemented. A document called *the specification of a computational aim* (or simply computational aim) contains this statement. A computational aim does not provide any details of *how* the mathematics will be implemented. An online database of computational aims [13] was developed as part of the TraCIM project (see section 7).

2. A computational aim is used as the basis to generate *reference data sets*. A reference data set consists of a number of *reference pairs*. A reference pair comprises reference input data and corresponding reference output data.

3. The software to be verified is presented with a selection of reference input data as test data. The output generated by the software is compared with the corresponding reference output data. If the values agree according to stated criteria (e.g. to a certain number of decimal digits) the software is deemed to be of the required quality.

## 4.2 Generating Reference Data Sets

In theory, reference data sets could be generated by someone reading the computational aim and calculating the reference pairs by hand! In practice, a *data generator* is implemented in software to generate reference pairs. Data generators can be either *static* or *dynamic*. Static generators produce a file of reference pairs; dynamic generators produce reference pairs on the fly.

Reference data sets can be generated in one of two ways:

- *Forward*: Begin with reference input data and, using reference software, generate corresponding reference output data.
- *Reverse*: Begin with the reference output data and generate corresponding reference input data. E.g. for minimum circumscribed circle calculations, described in section 7.3.2, define a circle by selecting a radius length and the ($x$, $y$) coordinates of the centre point. From these parameters, generate the ($x$, $y$) coordinates of a set of data points such that the specified circle is the smallest that contains all of these data points.

In general, reverse generation is easier to implement than forward generation and is the preferred technique. The mathematics underlying reference data set generation is beyond the scope of this paper; reference [4] provides further details.

# 5.0 Software Traceability

The term *traceability* is well established within software engineering. For example ISO/IEC/IEEE 24765:2010 [9] defines *requirements traceability* as "Discernible association between a requirement and related requirements, implementations, and verifications." Requirements traceability provides a means of validating software, i.e. ensuring requirements have been met.

In the context of this paper the authors mean *traceability* as providing evidence of numerical correctness through an unbroken chain ending with a computational aim via a reference data set (see figure 2, c.f. figure 1):

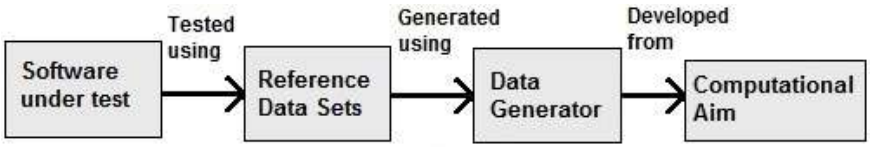


Figure 2: Software traceability chain

# 6.0 Coordinate Measuring Machines

Manufacturers of *coordinate measuring machines* are a major industrial user of reference data sets. A brief description of these devices and related standards will provide useful background information.

## 6.1 Overview of CMMs

Coordinate measuring machines (CMMs) [5] use probes to make measurements of the coordinates of points on the surface of an object. Mathematical algorithms implemented by software are used to infer information about the object (such as dimensions, positions of features, etc.). The object could have a complex geometry such as a camshaft or turbine blade.

As noted in [14], a complex engineering product (such as a gas turbine engine) can be constructed from tens of thousands of components. The importance of the accuracy *and consistency* of measurements made by devices such as CMMs cannot be overstated. If measurements made by one manufacturer's CMM differs from that of another manufacturer's, beyond an acceptable tolerance, then individual components might not fit together.

## 6.2 ISO 10360

International standards for testing the performance of CMMs help ensure consistency [14]. The ISO 10360 series of standards are among the most widely

adopted [15]. The majority of these standards are concerned with physical testing, such as the use of gauge blocks [16].

However, software that performs mathematical calculations (such as geometric element fitting) lies at the heart of CMMs. The quality of the CMM's software should be as important as the quality of the physical construction of the CMM itself. But how can the quality of such software be determined? CMM manufacturers may wish their software and algorithms implemented to remain confidential [17]. Reference data sets offer one means of testing.

ISO 10360-6:2001 [18] describes a procedure for testing software using reference data sets. To quote from the introduction to this standard "The reliability of information about features that is determined from associated features is influenced by the quality of the software for computing those features."

# 7.0 The TraCIM Project

"Traceability for Computationally Intensive Metrology" (TraCIM) [6] is an EU-funded project that ran from June 2012 to May 2015. *Traceability,* in the context of TraCIM, is described in section 5. *Computationally intensive metrology* refers to metrology applications that make significant use of mathematical software. Such applications include measurement devices such as CMMs.

The main aim of the project was to develop an information and communications technology (ICT) infrastructure for verifying mathematical software via the internet. This verification is traceable to computational aims via reference data sets maintained at NMIs. The work was undertaken by a consortium consisting of:

- The NMIs of the Czech Republic, Germany, Italy, the Netherlands, Poland, Slovenia and the UK.
- The universities of Huddersfield, Osfalia, York and Zwickau.

CMM manufacturers Hexagon, Mututoyo, Werth and Zeiss were unfunded industrial partners in the project. Their role was to provide guidance to ensure meeting the needs of industry was at the heart of TraCIM.

## 7.2 Work Packages

The work was divided into the following *work packages* (WPs):

- WP1: Framework for traceable computation in metrology
  The starting point for the project was to categorise various *metrology areas* (e.g. electricity and magnetism, length, mass and radiometry) and identify mathematical calculations relevant to those areas [19]. Calculations relevant to more than one metrology area are termed *interdisciplinary*.

Requirements for the ICT infrastructure were defined as part of WP1. The implementation was carried out in WP5.

- WP2: Formal statement of computational aims
  Software implementing a mathematical calculation can only be verified if there is a clear, complete and unambiguous statement of the mathematics. As stated in section 4.1, in the context of work summarised in this paper, these statements are called *computational aims*.

  The main aim of WP2 was to develop a selection of computational aims for mathematical calculations required by metrology areas identified in WP1. A common template for these computational aims was developed and a searchable database made available online [13].

- WP3: Generation of reference data
  In this WP, data generators were developed for a selection of computational aims identified in WP1 and WP2.

- WP4: Performance metrics
  In this WP, metrics were designed to evaluate the performance of software under test. E.g.:
    - How close is the reference data set to the true mathematical solution of the computational aim? Knowing the quality of the reference data set is a part of knowing the quality of the software under test [20].
    - Define a maximum permissible error (MPE) that applies in the relevant metrology domain.

- WP5: Launch of TraCIM System
  In this WP, the ICT infrastructure whose requirements were drawn up as part of WP1 was developed and demonstrated for the unfunded partners of the project.

Work packages were divided into *deliverables*. For the remainder of this section the authors provide further details of those deliverables most relevant to this paper.

## 7.3 WP2: Formal Specification of Computational Aims

Formal methods [21], such as Z [22], bring mathematical rigour to software specification. A point that arises almost immediately is "If metrologists use mathematics to document computational aims, how can you get more formal than that?" Computational aims are indeed heavily mathematical documents. However:

- Omissions and ambiguities may occur, even in computational aims expressed using mathematical notation; would the use of formal methods allow these omissions and ambiguities to be identified and addressed?

- Would the added discipline that formal methods bring allow *better*, more clearly thought out, computational aims to be written?
- Specifications using formal methods can be analysed using software tools.

The University of York was awarded a one-year research grant to explore the use of formal methods for the specification and analysis of computational aims. The following subsections summarise the research. Anyone interested in further details of the formal specification work undertaken as part of TraCIM is welcome to contact the authors of this paper.

### 7.3.1 Initial Review

The first stage was to decide which formal specification language to use. Both VDM [23] and EXPRESS [24], which has received attention in the field of metrology, were considered. However, Z was selected, as its expressive style is closest to the mathematics used to write computational aims; the other languages considered use more software-oriented constructions. Z is supported by software tools that provide syntax and type checking, as well as other features [25]. An ISO standard has also been defined for Z [26].

### 7.3.2 Z Specification: Minimum Circumscribed Circle (MCC)

The next stage was to apply Z to specifying some existing computational aims. The first computational aim chosen was "Minimum circumscribed circle (MCC) to data in the *xy*-plane" (search [11] for "minimum circumscribed circle"). Being straightforward, this computational provides a good starting point for exploring the use of formal methods.

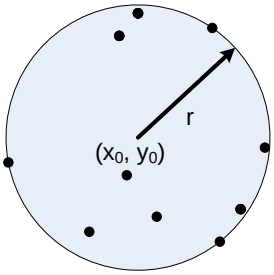


Figure 3. Minimum Circumscribed Circle

The informal description of this computational aim is "Determine centre coordinates and radius of the circle of minimum radius that circumscribes a given set of points in the *xy*-plane."; see figure 3.

Z specifications are structured using *schemas*. The upper section of a schema contains variable declarations; the lower section defines the relationship between the values of the variables and any constraints on these values. Schemas can be named and combined with other schemas.

Schema *MCCInputs*, defined below, specifies the inputs to the computation:

- $M$, the number of data points
- $X$, a matrix ($m \times 2$) containing the coordinates of those data points; $m$ is constrained to be a non-zero natural – i.e. the set of data points must not be empty

*MCCInputs*

$m : \mathbb{N}_1$

$X : \mathbb{M}$

---

$X \in \textbf{realmatrix}(m, 2)$

Schema *MCCOutputs* specifies the outputs of the computation. These outputs are:

- $X_0$, a vector containing two real numbers which are the ($x$, $y$) coordinates of the centre of the MCC
- $r$, a real number that is the radius of the MCC. $r$ is constrained to be $\geq 0$

*MCCOutputs*

$X_0 : \textbf{realvector}(2)$

$r : \mathbb{R}$

---

$r \geqslant 0$

Schema *MCCComputation* completes the specification of the computational aim. Including the name of other schemas in the upper section of a schema imports all variable declarations, constraints etc. from those schemas.

*MCCComputation*

*MCCInputs*

*MCCOutputs*

---

$\langle X_0(1), X_0(2), r \rangle = \textbf{safemin}_v($

$\{x_0, y_0, r : \mathbb{R} \mid$

$(\forall i : 1 .. m \bullet (X(i)(1) - x_0)^2 + (X(i)(2) - y_0)^2 - r^2 \leqslant 0)$

$\bullet (\langle x_0, y_0, r \rangle, r) \}$ )

- $<x_0, y_0, r>$ is a sequence of values that define the $x$ and $y$ coordinates and radius of a circle containing the data points held in matrix $X$
- $X(i)(1)$ and $X(i)(2)$ where $i = 1 .. m$ contain the ($x$, $y$) coordinates of the data points
- The function $\textbf{safemin}_v$ inputs a set of pairs ($<x_0, y_0, r>$, $r$) and returns the value of $<x_0, y_0, r>$ where $r$ is a minimum

The definitions of $\textbf{safemin}_v$, **realvector** and **realmatrix** are not included in this paper due to space constraints.

### 7.3.3 Analysis of Computational Aims

Confidence in the validity of formal specifications can be increased by analysis using software tools. Mathematica [27] is a symbolic (and numerical) workbench supporting mathematical reasoning. It allows users to simplify formulae symbolically, via its **FullSimplify** function. Formulae that can be simplified include those that yield logical values.

**FullSimplify** can be used to confirm whether a formal specification has required properties. For example, the MCC for an input data set containing two (distinct) data points is the circle which has a diameter defined by those points. This property can be characterised in Z as:

$$\vdash? \forall\, MMCComputation \mid m = 2 \bullet Circle((X_0(1), X_0(2)), r) = DiamToCircle(Line(X((1)(1), X(1)(2)), (X(2)(1), X(2)(2))))$$

This property is then characterised in Mathematica as:

```
PropertyDiag[x1_, y1_, x2_, y2_] :=
       TwoPointMCCCircle[x1, y1, x2, y2] ==
       DiameterLinetoCircle[Line[{{x1, y1}, {x2, y2}}]]
```

where the definition of **TwoPointMCCCircle** contains the variable declarations, constraints etc. from *MCCComputation*. With the addition of some further functions, not discussed here for space reasons, **FullSimplify** returns `True`.

## 7.4 WP5: The TraCIM System

The TraCIM System [7] delivers traceable verification, by means of software testing, directly to systems containing the software under test. A web interface [28] allows service users to register with the system and order test(s). Each test relates to a particular computational aim. The service user will be emailed an order key that allows access to the test(s).

The TraCIM System consists of the following components:

- *The TraCIM Server*: The core software module that controls the flow of data to and from other modules. Typically the TraCIM Server is hosted by an NMI
- *Expert extensions*: For each computational aim, a software module called an *expert extension* implements all operations relating to that aim. As will be discussed, these operations include selecting reference pairs.
- *TraCIM Client*: This software module connects the software under test to the TraCIM Server, via the internet.

As illustrated in figure 4, a TraCIM Client requests some test data using the order key. In this example a client running within a CMM is used to contact the TraCIM Server via the internet.

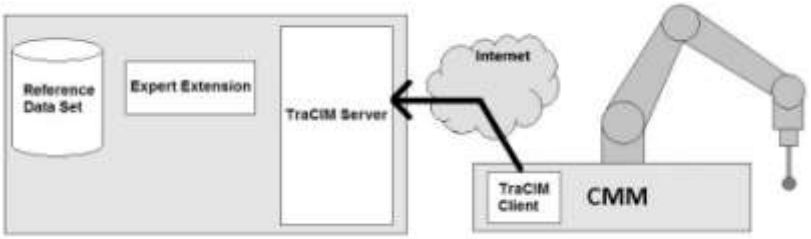


Figure 4. Request test data

As illustrated in figure 5, the TraCIM Server calls up the relevant expert extension which makes a selection of reference pairs from the reference data set. Reference input data from the pairs are supplied to the software under test (ST):

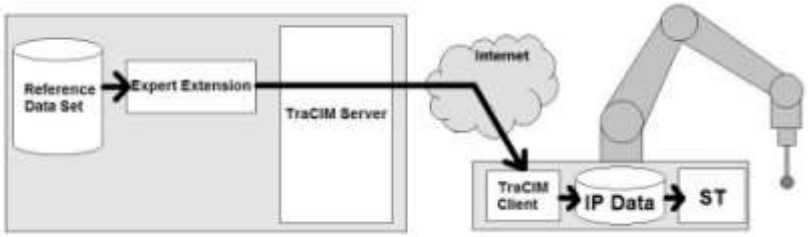


Figure 5. Obtain test data and input to software under test

As illustrated in figure 6, the software under test processes the reference input data and generates output data. The TraCIM Client supplies the output data to the TraCIM Server:

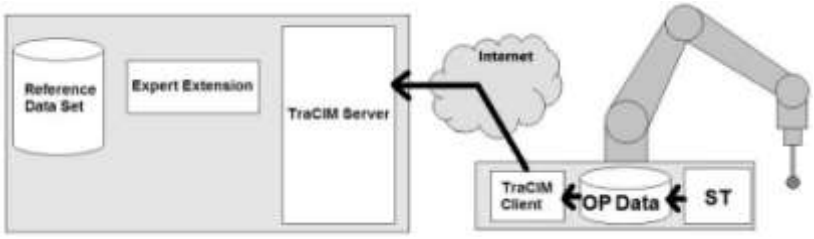


Figure 6. Software under test produces output data

As illustrated in figure 7, the TraCIM Server supplies the output data generated by the software under test to the expert extension; this data is then compared to reference output data. If the output data and reference output data agree according to the required criteria (e.g. to a certain number of decimal digits) the software is deemed to have *passed* (i.e. is of the required quality); otherwise the software will

have *failed*. The TraCIM Server sends a test report to the TraCIM Client and the client outputs a PDF file:

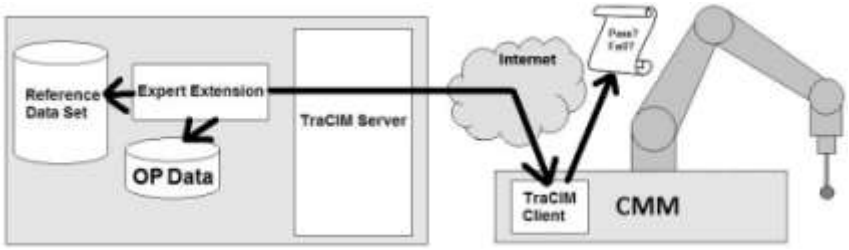


Figure 7. Compare results and generate report

# 8.0 The ValTraC project

The "Validation of software development and analysis tools using TraCIM" (ValTraC) project is another EU-funded project; it is currently underway, running from July 2016 to December 2017.

ValTraC explores how the TraCIM System can be used to verify mathematical software from other application areas in addition to physical measurement systems such as CMMs. With guidance from industry, further expert extensions are being developed for the TraCIM System. The work is being undertaken by NPL, PTB and Osfalia University.

With reference to section 2, this project should have been named VerTraC.

# 9.0 Further Thoughts

TraCIM provides an infrastructure that allows the concept of metrological traceability to be extended to the verification of mathematical software. The key concept is a *traceability chain* that links the software under test to a computational aim. Further research could add greater formality to this chain. For example:

- The results of the work within TraCIM on using formal methods to define computational aims were encouraging and demonstrated there is further research that could be carried out. For example, could refinement [29] allow formal methods to extend a chain of traceability from a reference data set to a computational aim in a formal, documented manner? Would such an extension bring any benefit?

- Functional programming languages such as Haskell [30] evaluate expressions rather than assign values to variables. Mathematical libraries are being developed for Haskell [31]. Developing data generators using Haskell would be an interesting test of those libraries.

It is hoped that this paper will make a useful contribution to such research being undertaken.

## 10.0 Acknowledgements

This work has been carried out as part of the European Metrology Programme for Innovation and Research (EMPIR) project 15SIP06. The EMPIR initiative is co-funded by the European Union's Horizon 2020 research and innovation programme and the EMPIR Participating States.

The research on formal methods within TraCIM was undertaken by Andy Galloway and Richard Paige of the Department of Computer Science at the University of York. Section 7.3 was adapted from reports written for TraCIM by Andy Galloway.

The authors would like to thank our colleagues Peter Harris for reviewing this paper and Stuart Davidson for his assistance in writing section 3

## 11.0 References

1 National Physical Laboratory. Retrieved 7th March 2017 from National Physical Laboratory: http://www.npl.co.uk/
2 Physikalisch-Technische Bundesanstalt. Retrieved 7th March 2017 from Physikalisch-Technische Bundesanstalt: https://www.ptb.de/
3 Berthold J, Imkamp D, Looking at the future of manufacturing metrology: Roadmap document of the German VDI/VDE Society for Measurement and Automatic Control. Retrieved 7th March 2017 from Journal of Sensors and Sensor Systems: http://www.j-sens-sens-syst.net/2/1/2013/jsss-2-1-2013.pdf
4 Cox M, Harris P, The design and use of reference data sets for testing scientific software. Retrieved 7th March 2017 from ResearchGate: https://www.researchgate.net/publication/222491831_Design_and_Use_of_Reference_Data_Sets_for_Testing_Scientific_Software
5 Coordinate Measuring Machines. Retrieved 7th March 2017 from Mitutoyo: http://www.mitutoyo.co.uk/coordinate-measuring-machines
6 TraCIM project website. Retrieved 7th March 2017 from tracim.eu: http://www.tracim.eu/
7 Härtig F, Tang J, Hutzschenreuter D, Wendt K, Kniell K, Shi Z, Online validation of comparison algorithms using the TraCIM-system, Int. J. Mech. Eng. Autom. Volume 2, Number 7, 2015, pp. 312-327
8 European Metrology Programme for Innovation and Research (EMPIR) Retrieved 7th March 2017 from EMPIR https://www.euramet.org/research-innovation/research-empir/empir-calls-and-projects/empir-call-2015/
9 ISO/IEC/IEEE 24765:2010 Systems and software engineering — vocabulary.
10 International Bureau of Weights and Measures (BIPM). Retrieved 7th March 2017 from BIPM: http://www.bipm.org/
11 International Vocabulary of Metrology (VIM). Retrieved 7th March 2017 from BIPM: http://www.bipm.org/en/publications/guides/vim.html
12 Bell S, A beginner's guide to uncertainty of measurement. Retrieved 7th March 2017 from NPL: http://www.npl.co.uk/publications/a-beginners-guide-to-uncertainty-in-measurement

13 TraCIM: Computational Aims Database. Retrieved 7th March 2017 from National Physical Laboratory: http://www.tracim-cadb.npl.co.uk
14 Saunders P, Wilson A, Orchard N, Tatman d N, Maropoulos P, An exploration into measurement consistency on coordinate measuring machines. Retrieved 7th March 2017 from Science Direct: http://www.sciencedirect.com/science/article/pii/S2212827114010361
15 Performance of CMMs: Testing, Calibration and Uncertainty. Retrieved 7th March 2017 from Mitutoyo. http://www.mitutoyo.com/news/resource-center/performance-of-cmms-testing-calibration-and-uncertainty/
16 ISO 10360-2:2009 Geometrical product specifications (GPS) — Acceptance and reverification tests for coordinate measuring machines (CMM) Part 2: CMMs used for measuring linear dimensions (ISO 10360-2:2009).
17 Sung J. A, Least Squares Orthogonal Distance Fitting of Curves and Surfaces in Space, Springer-Verlag Berlin Heidelbert 2004, ISBN 0302-9743, Pages 14 - 16
18 ISO 10360-6:2001 Geometrical product specifications (GPS) – Acceptance and reverification tests for coordinate measuring machines (CMM) – Part 6: Estimation of error in computing Gaussian associated features
19 TraCIM applications. Retrieved 7th March 2017 from tracim.eu http://www.tracim.eu/emrp/fileadmin/documents/tcim/TraCIM_-_Database_of_applications.pdf
20 Kok G. J. P, Harris P. M, Smith I. M, and Forbes A. B. (2016, July) Reference data sets for testing metrology software Metrologia Volume 53, pp. 1091-1100, doi: 10.1088/0026-1394/53/4/1091.
21 Woodcock J, Loomes M, Software Engineering Mathematics. Pitman, 1988
22 Spivey J. M, The Z Notation: A reference manual (2nd ed). Prentice Hall, 1992
23 Jones C. B, Systematic Software Development using VDM. Prentice Hall. 1990
24 ISO 10303-11:2004 Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual
25 Community Z Tools. Retrieved 7th March 2017 from sourceforge.net http://czt.sourceforge.net/
26 ISO/IEC 13568 Information technology — Z formal specification notation — Syntax, type system and semantics. First edition 2002-07-01
27 Wolfram Mathematica. Retrieved 7th March 2017 from Wolfram https://www.wolfram.com/mathematica/
28 PTB TraCIM System. Retrieved 7th March 2017 from PTB: https://tracim.ptb.de/
29 Woodcock J, Davies J, Using Z: Specification Proof Refinement, Prentice Hall, 1996
30 The Haskell Programming Language. Retrieved 7th March 2017 from Haskell.org https://wiki.haskell.org/Haskell
31 The Haskell Programming Language / Applications and libraries / Mathematics. Retrieved 7th March 2017 from Haskell.org https://wiki.haskell.org/Applications_and_libraries/Mathematics